Развој интелигентног агента за игру Отело користећи Deep Q Learning

1. Назив теме

Развој аутономног агента за игру Отело (Reversi) на табли 8×8 користећи Deep Q Network (DQN) алгоритам

2. Дефиниција проблема

Проблем који се рјешава јесте развој интелигентног агента који може да игра стратешку игру Отело (Реверси) на професионалном нивоу. Отело игра за два играча са потпуним информацијама гдје је циљ имати више фигура своје боје на табли када се игра заврши. Агент треба да научи оптималне стратегије играња кроз интеракцију са окружењем користећи технике дубоког појачаног учења, конкретно Deep Q Network алгоритам.

3. Мотивација за проблем који се рјешава

Развој агената за игре као што је Отело важан је јер пружа платформу за истраживање техника појачаног учења које се касније могу примијенити у ширем контексту. Модели засновани на DQN-у примјењиви су у:

- **роботици** (доношење одлука у непредвидивим окружењима),
- **оптимизацији ресурса** (енергија, саобраћај, логистика),
- **обучавању система за стратешко планирање** (системи препорука, аутономна вожња).

Игре као Отело служе као добро контролисана лабораторија у којој се могу тестирати и унапређивати технике које имају потенцијалну друштвену корист.

4. Скуп података

Користиће се скуп од **25 649 партија Отела** преузет са Kaggle платформе:

- **Извор**: https://www.kaggle.com/datasets/andrefpoliveira/othello-games/data
- **Опис**: Скуп садржи све стандардне игре које су играли ТОП 100 играча на eOthello платформи
- **Садржај**: Без насумичних отварања, Anti варијанти, Hexa или Grand режима
- **Атрибути**: Сваки запис садржи секвенцу потеза, резултат партије, оцјене играча
- **Циљно обиљежје**: За надгледано предтренирање - оптималан потез у датој позицији

5. Начин претпроцесирања података

- **Конверзија партија** у секвенце стања табле и одговарајућих потеза
- **Аугментација података** кроз ротације и рефлексије табле (8 симетрија)
- **Кодирање стања** у тензоре димензија 8×8×N гдје је N број улазних слојева:
    - Слој 1: Црне фигуре (1 ако постоји црна фигура, иначе 0)
    - Слој 2: Бијеле фигуре (1 ако постоји бијела фигура, иначе 0)
    - Слој 3: Легални потези (1 ако је потез легалан, иначе 0)
    - Слој 4: Тренутни играч (1 ако је на потезу, иначе 0)
- **Додатни слојеви** (опционо): историја претходних потеза, стабилне фигуре…

6. Методологија

6.1 Deep Q Network (DQN) алгоритам

DQN комбинује Q-learning са дубоким неуронским мрежама. Агент учи Q-функцију Q(s,a) која процјењује вриједност извршавања акције 'a' у стању 's'. Кључне компоненте:

- **Experience Replay**: чување искустава у буферу за стабилније учење
- **Target Network**: одвојена мрежа за стабилније циљне вриједности
- **ε-greedy стратегија**: баланс између истраживања и искоришћавања

6.1.1 Стања и акције

Стање представља једнан распоред фигура на табли уз информацију о томе ко је на потезу. Скуп акција из неког стања представља скуп свих легалних потеза према правилима игре Отело.

6.2 Архитектура неуронске мреже

- **3 конволутивна слоја**: за детекцију просторних образаца на табли
    - Conv1: 32 филтера 3×3, ReLU активација
    - Conv2: 64 филтера 3×3, ReLU активација
    - Conv3: 64 филтера 3×3, ReLU активација
- **2 потпуно повезана слоја**: за комбиновање карактеристика
    - FC1: 512 неурона, ReLU активација
    - FC2: 64 излаза (по један за свако поље табле)

6.3 Процес учења

1. **Иницијализација** DQN мреже и target мреже
2. **Self-play тренирање** против случајних противника и претходних верзија
3. **Experience collection** током игара
4. **Batch учење** из replay буфера

5. **Периодично ажурирање** target мреже

6.4 Потенцијални проблеми

- **Спора конвергенција** због велике сложености стања
- **Нестабилност учења** усљед промјенљиве стратегије противника
- **Катастрофално заборављање** претходно научених стратегија

6.5 Побољшања основног алгоритма

Отело је игра са изузетно великим простором стања (фактор гранања ~10, типичан број потеза ~60), те је стога разумно очекивати да основна имплементација DQN алгоритма неће дати задовољавајуће резултате. Стога ће бити имплементирана сљедећа побољшања:

- **Додатни улазни слојеви**: претходни потези, стабилност, мобилност…
- **Хеуристичке награде**: бонуси за контролу углова и ивица
- **Actor-Critic архитектура**: паралелно учење стратегије и вриједносне функције
- **Supervised pre-training**: иницијално тренирање на скупу експертских партија
- **Monte Carlo Tree Search**: комбиновање са MCTS-ом за боље планирање

За детаљније објашњење наведених техника погледати литературу.

7. Начин евалуације

- **Подјела података**: 80% тренирање / 10% валидација / 10% тестирање (за надгледано предтренирање)
- **Self-play евалуација**: тестирање против различитих стратегија (случајни, greedy, претходне верзије)
- **Метрике**:
    - Проценат побједа против базних стратегија
    - Конвергенција Q-вриједности током тренирања

7.1 Стратегије за поређење

Да би се евалуација резултата агента објективно извршила, потребно је имати референтне стратегије уз које ће се мјерити напредак. У овом пројекту користиће се сљедећи baseline примјери:

1. **Random agent** – агент који бира легалан потез насумично. Служи као доња граница учинка.
2. **Greedy agent** – агент који у сваком кораку бира потез који тренутно максимизира број освојених фигура, без дугорочног планирања. Ово је стандардни baseline у многим радовима о Отелу.

3. **Heuristic agent** – једноставна стратегија заснована на фиксним хеуристикама (статичка евалуација вриједности поља на табли, хеуристике мобилности, стабилности…).
4. **Supervised baseline** – модел обучен искључиво на Kaggle скупу експертских партија без појачаног учења. Ово омогућава поређење DQN-а са чисто надгледаним приступом.

Овај приступ је у складу са референтним радовима о DQN агентима у играма попут Отела и Го-а (Silver et al., 2016; Wennborg, Othello AI Exercise).

8. Технологије

- **Python 3.10+**: основни програмски језик
- **PyTorch**: за имплементацију неуронских мрежа и DQN алгоритма
- **Rust**: за game engine користећи bitboard репрезентацију због перформанси
- **NumPy**: за нумеричке операције
- **Matplotlib**: за визуелизацију резултата тренирања
- **PyO3**: за интеграцију Python-Rust кода

9. Литература

- **World Othello Federation**, Official Rules for the Game of Othello
- **Mnih, V., et al. (2015)**. "Human-level control through deep reinforcement learning." Nature, 518, 529-533.
- **Silver, D., et al. (2016).** “Mastering the game of Go with deep neural networks and tree search”, Nature, 529, 484-489
- **Silver, D., et al. (2017)**. "Mastering the game of Go without human knowledge." Nature, 550, 354-359.
- **Konda, V. R., & Tsitsiklis, J. N. (2000).** "Actor-critic algorithms." NIPS
- **Van Hasselt, H. et al. (2016).** “Deep Reinforcement Learning with Double Q-learning”, arXiv:1509:06461
- Bitboards - Chess Programming WIKI
- **Hans Wennborg**, Othello for Desktop, Mobile and Web: an AI and GUI Exercise
- Othello Games Dataset